# Evolutionary Design Of Complex Software (EDCS)

John Salasin, Ph.D.

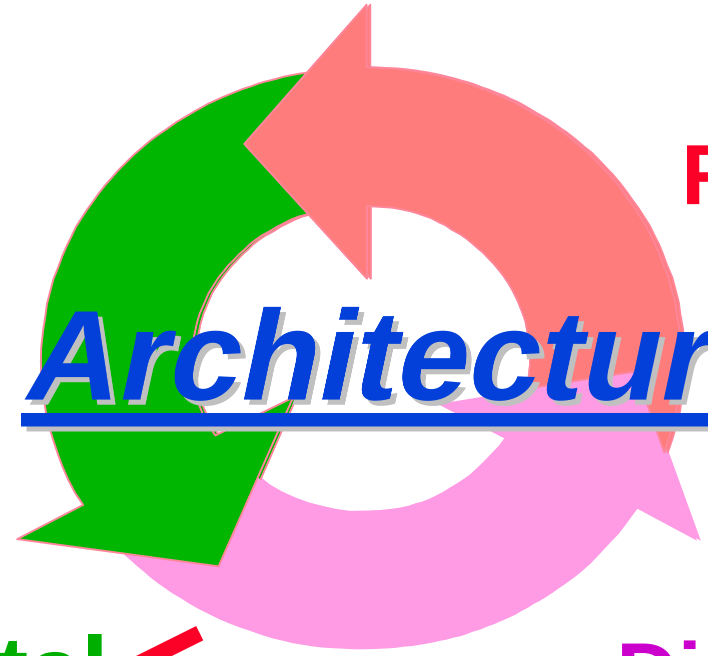
# The Problem

- **DoD depends on large, complex software systems**
- **These systems must evolve over time**
- **Today: Even small software changes can have unintended and far reaching effects**
  - Global implications must be analyzed
  - Entire system must be retested and recertified
- **Impact of present approach:**
  - Change is slow, error prone, and cumbersome
  - Cost of change is proportional to size of system (or worse), not size of change

# EDCS Technical Approach

**Design Management**

**Test and Recertification**



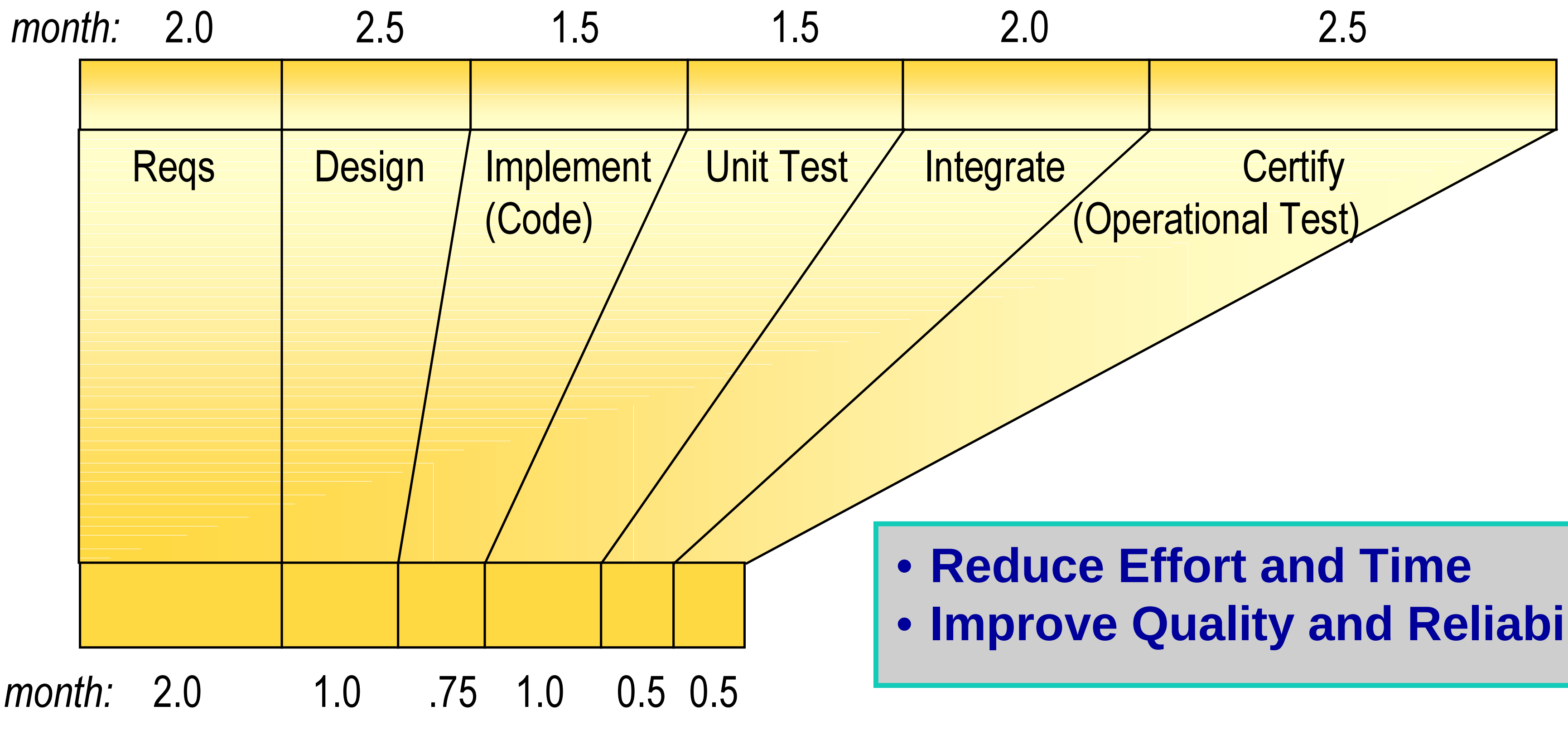
***Architecture***

**Disciplined Construction**

~~**Incremental change incurs incremental cost!**~~

# B-2 Software Release Cycle Example

*Cut Cycle Time in Half*



- **Reduce Effort and Time**
- **Improve Quality and Reliability**

# EDCS Life Cycle Cost Savings

| Activity | % of effort | % savings | Approach |
|---|---|---|---|
| Archeology | 50 | 90 | Use of standard architectures and notations, design reuse, rapid prototyping |
| Requirements/ Analysis | 6 | 20 | Standard architectures, analysis reuse, rapid prototyping |
| Design | 6 | 50 | Design and analysis reuse |
| Code | 6 | 95 | Code reuse, automated generation / composition |
| Test | 12 | 90 | Incremental testing, Formal Methods (“correct by construction”) |
| Redo | 10 | 75 | Automated generation/composition |
| Documentation | 10 | 95 | Standard notations are “self documenting” |

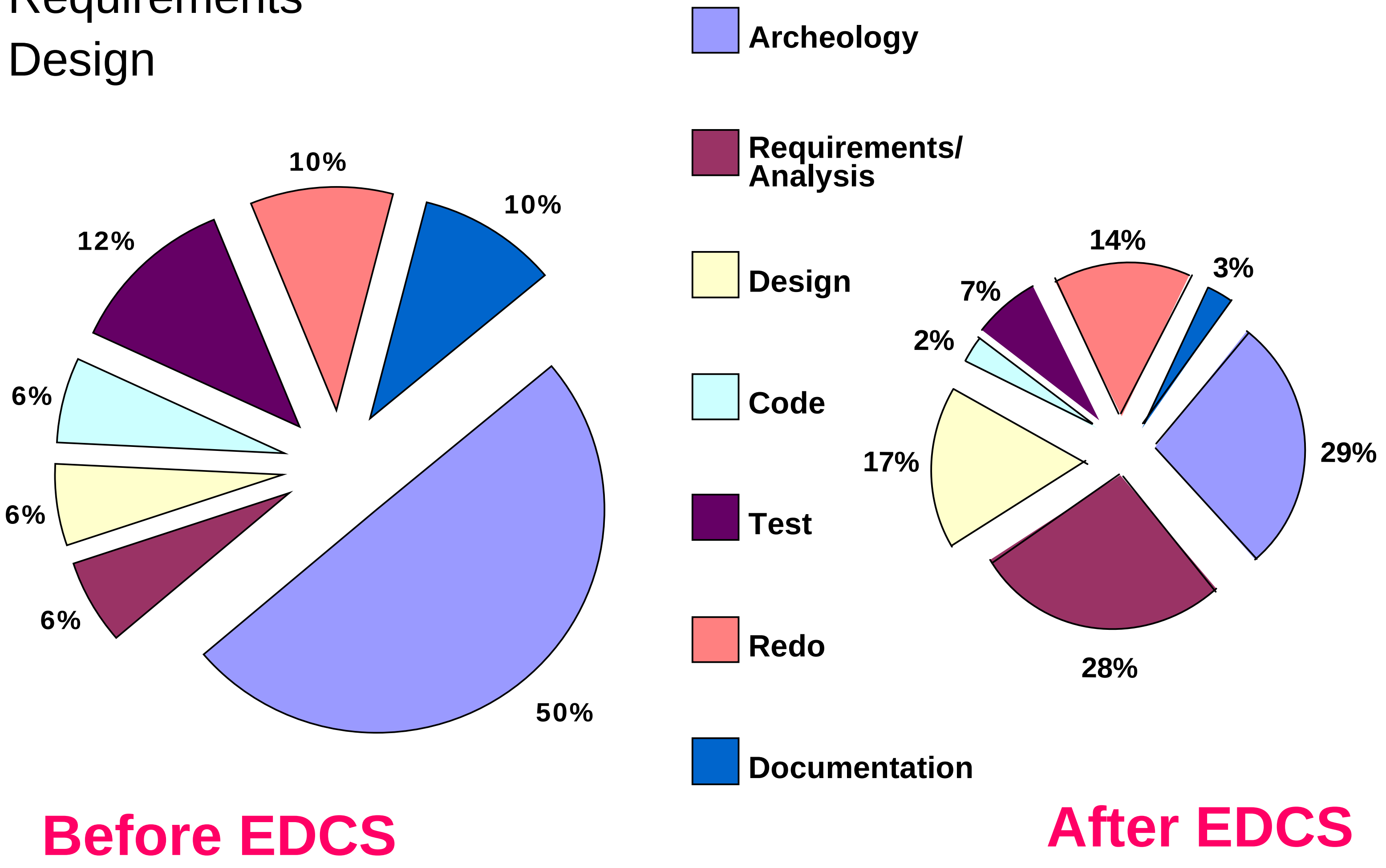
**Total savings objective 80%+**

*Approved for Public Release, Distribution Unlimited*

# Cost Saving and Shifting

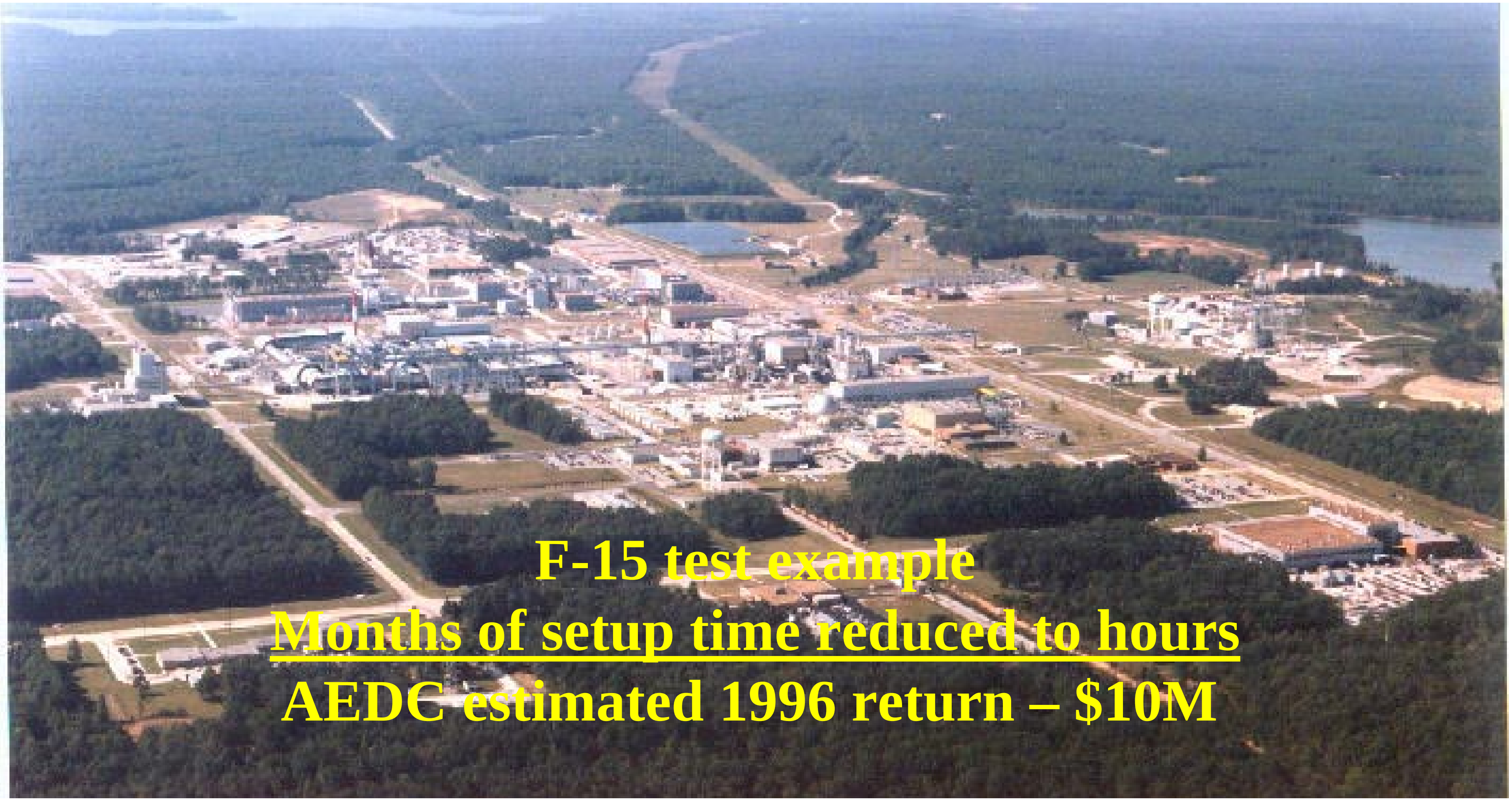
80% Overall Reduction (circles not to scale)

- Proportionally more “thinking” time
  - Requirements
  - Design



# Shortening Cycle Time

Package



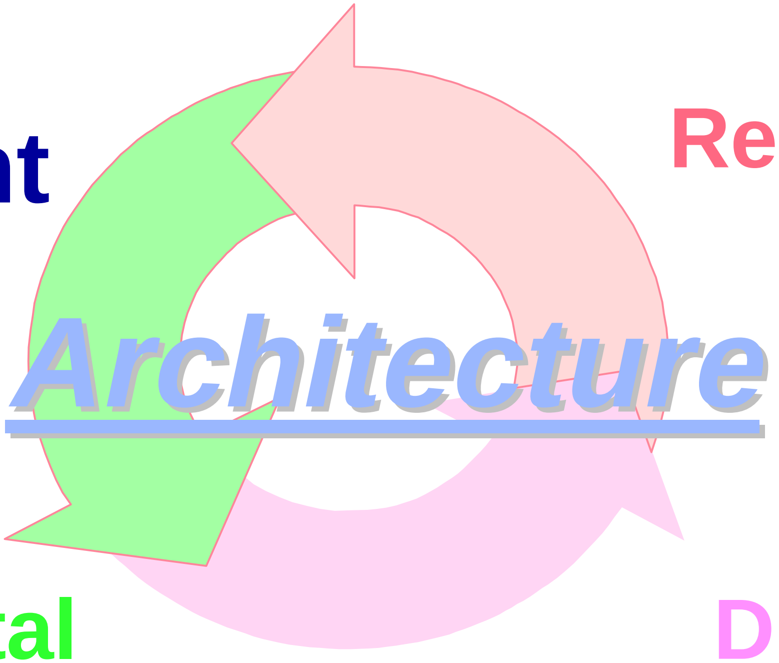
F-15 test example

<u>Months of setup time reduced to hours</u>

AEDC estimated 1996 return – $10M

Approved for Public Release, Distribution Unlimited

# EDCS Technical Approach



**Design Management**

**Test and Recertification**

***Architecture***

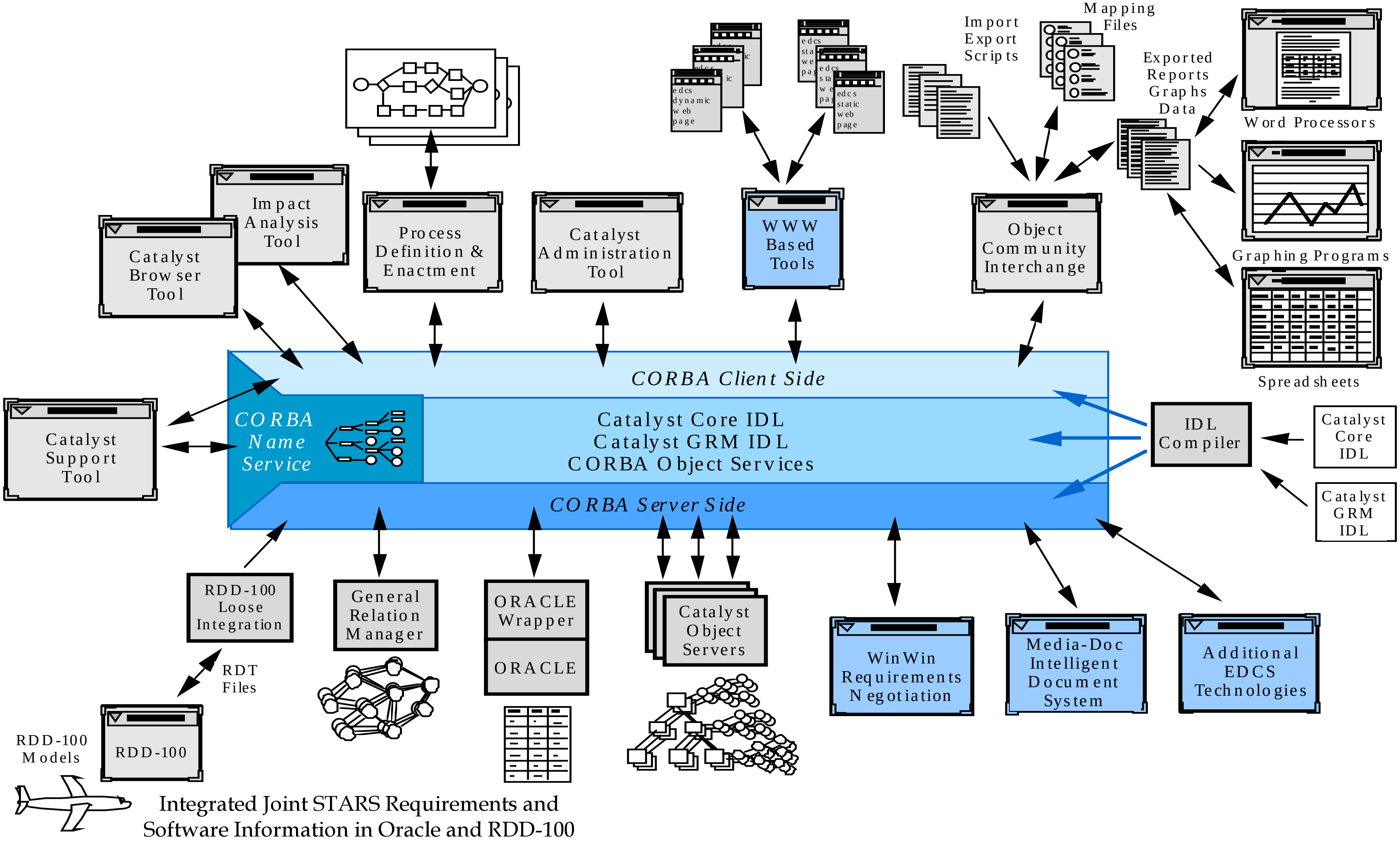
**Incremental change incurs incremental cost!**

**Disciplined Construction**

# Design Management

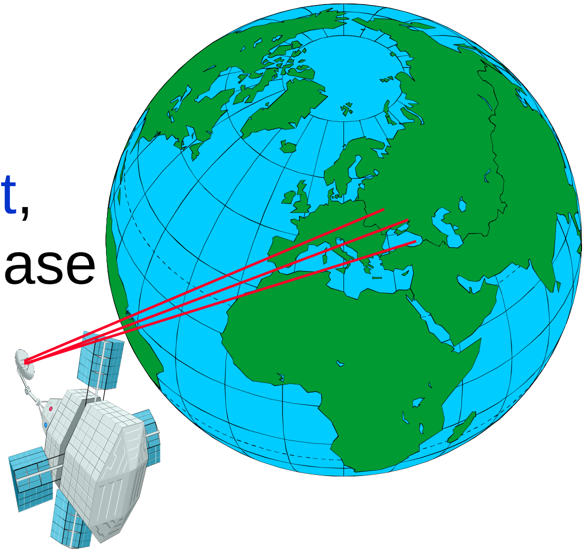
- **Requirement:**
  - Rapid identification of components affected by a proposed change
- **Today:**
  - System level design rationale not captured
    - **Design-time analyses are lost**
  - Component design information is dispersed
    - **Difficult to access**
    - **Not consistent**
- **EDCS Approach**
  - Design Web
  - Automate capture of design rationale

# Design Web


Impact Analysis Tool
Catalyst Browser Tool
Process Definition & Enactment
Catalyst Administration Tool
edcs dynamic web page
edcs static web page
WWW Based Tools
Import Export Scripts
Mapping Files
Exported Reports Graphs Data
Object Community Interchange
Word Processors
Graphing Programs
Spreadsheets
CORBA Client Side
CORBA Name Service
Catalyst Core IDL
Catalyst GRM IDL
CORBA Object Services
CORBA Server Side
Catalyst Support Tool
IDL Compiler
Catalyst Core IDL
Catalyst GRM IDL
RDD-100 Loose Integration
RDT Files
RDD-100 Models
RDD-100
General Relation Manager
ORACLE Wrapper
ORACLE
Catalyst Object Servers
WinWin Requirements Negotiation
Media-Doc Intelligent Document System
Additional EDCS Technologies
Integrated Joint STARS Requirements and Software Information in Oracle and RDD-100

# What's Exciting: Design Management

- Collaboration environment supporting rationale capture, management and evolution
- Methodology and tools to develop mission scenarios and issue-based analyses of alternatives
- Reverse engineering: Automated extraction of behavior and structure
- Generation of design explanations specific to individual user tasks and needs
- Access to legacy design databases
- Distributed Configuration Management, including versioning and software release management

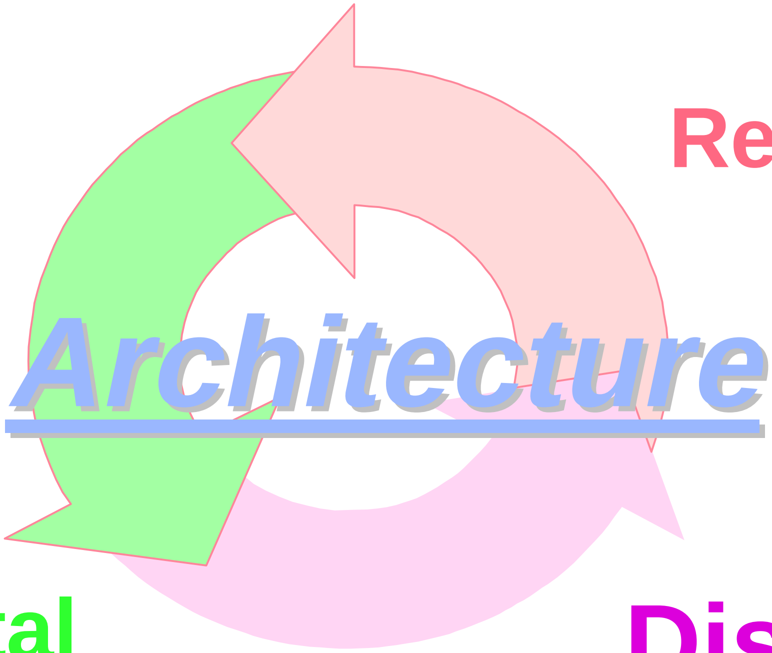


Approved for Public Release, Distribution Unlimited

# EDCS Technical Approach

Design Management

Test and Recertification



**Architecture**

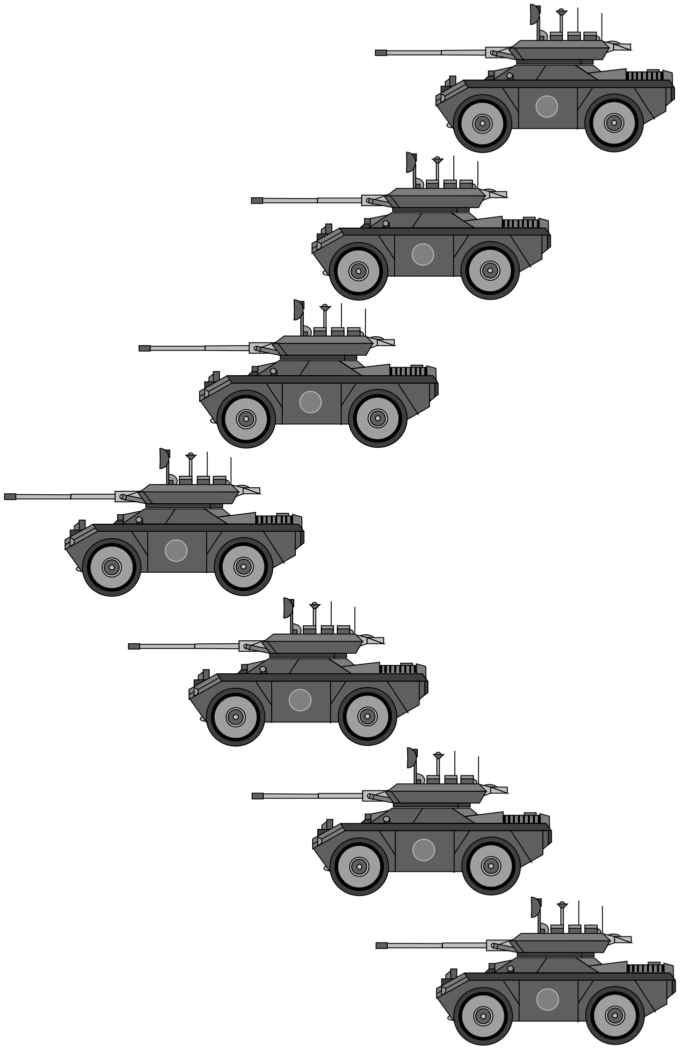
**Incremental change incurs incremental cost!**

Disciplined Construction

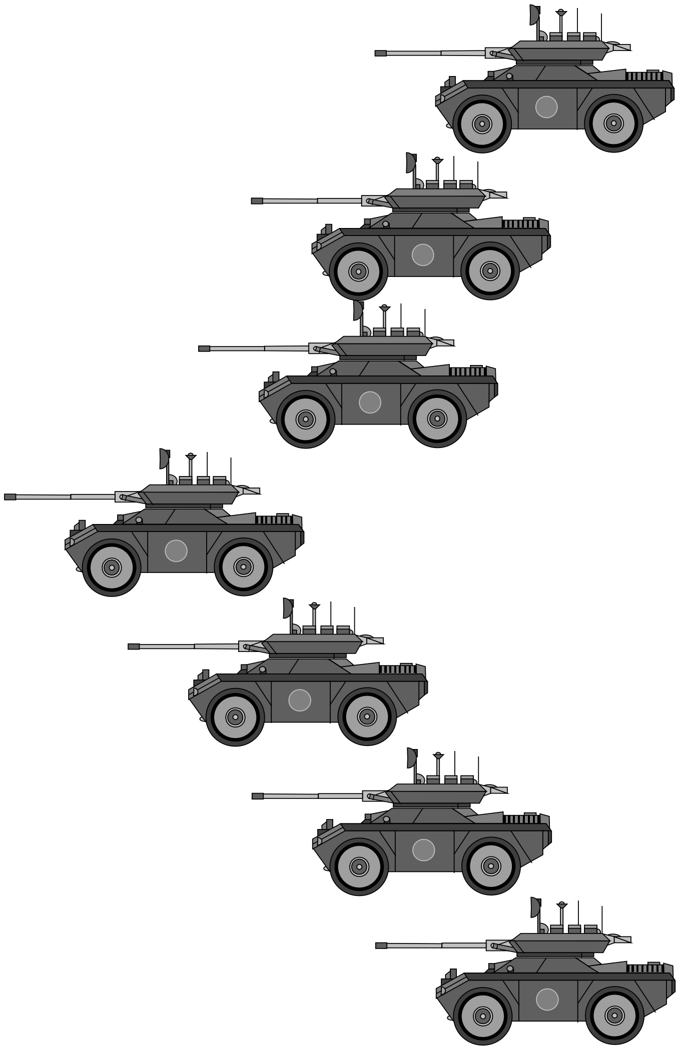
# Disciplined Construction

- **Requirement – Ability to make isolated changes without chaotic effects**
- **Problem today**
  - Lack rigorous means to isolate components and analyze interactions
  - Programmers make mistakes
- **Approach**
  - New technologies to specify / analyze components and their interactions
  - Automated composition and generation of code from specifications

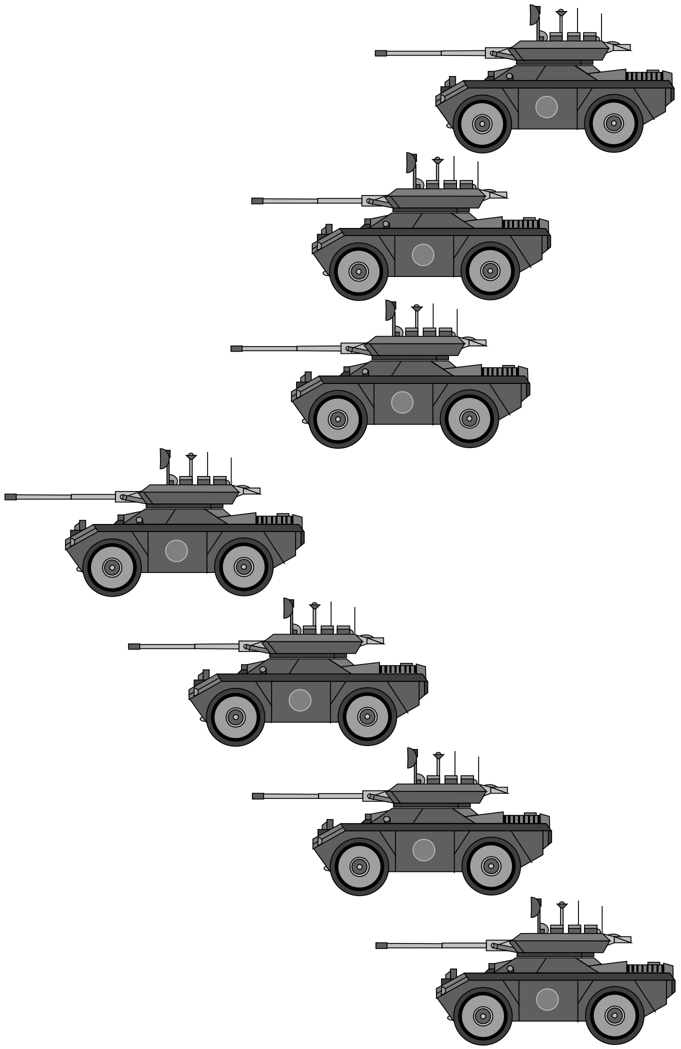
# What is Wrong With This Simulated Platoon?



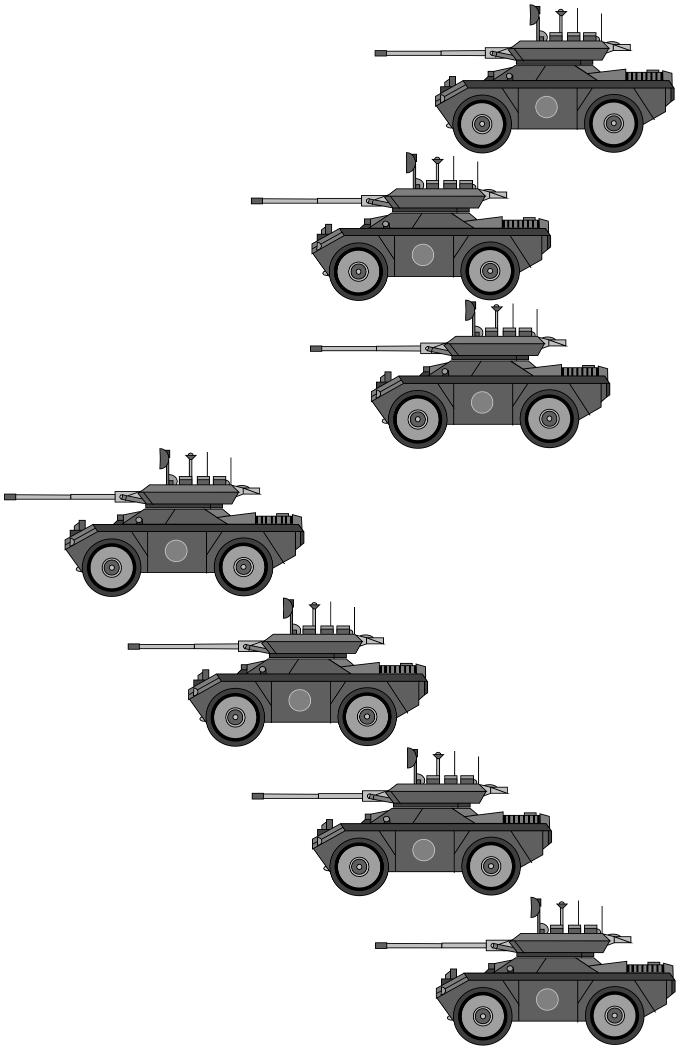
# What is Wrong With This Simulated Platoon?



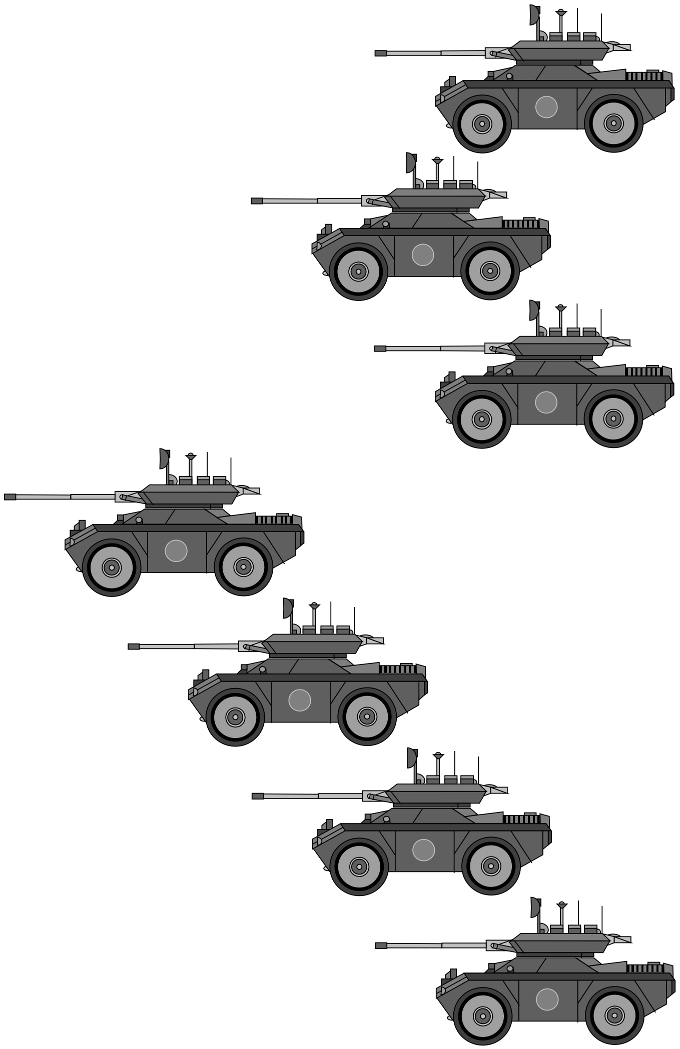
# What is Wrong With This Simulated Platoon?



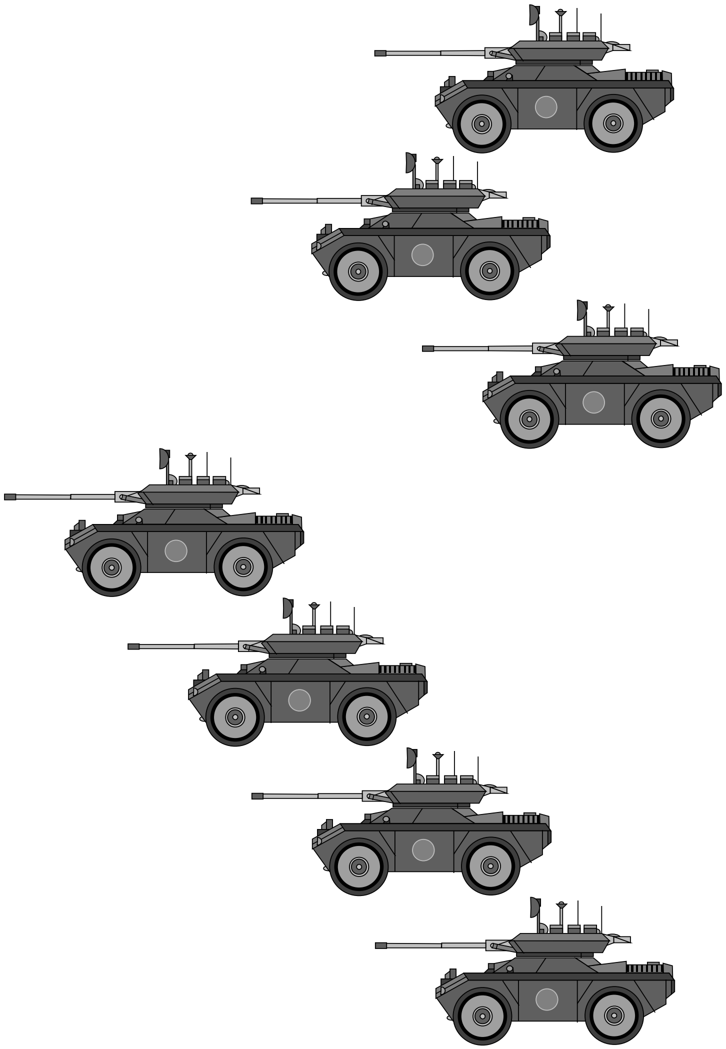
# What is Wrong With This Simulated Platoon?



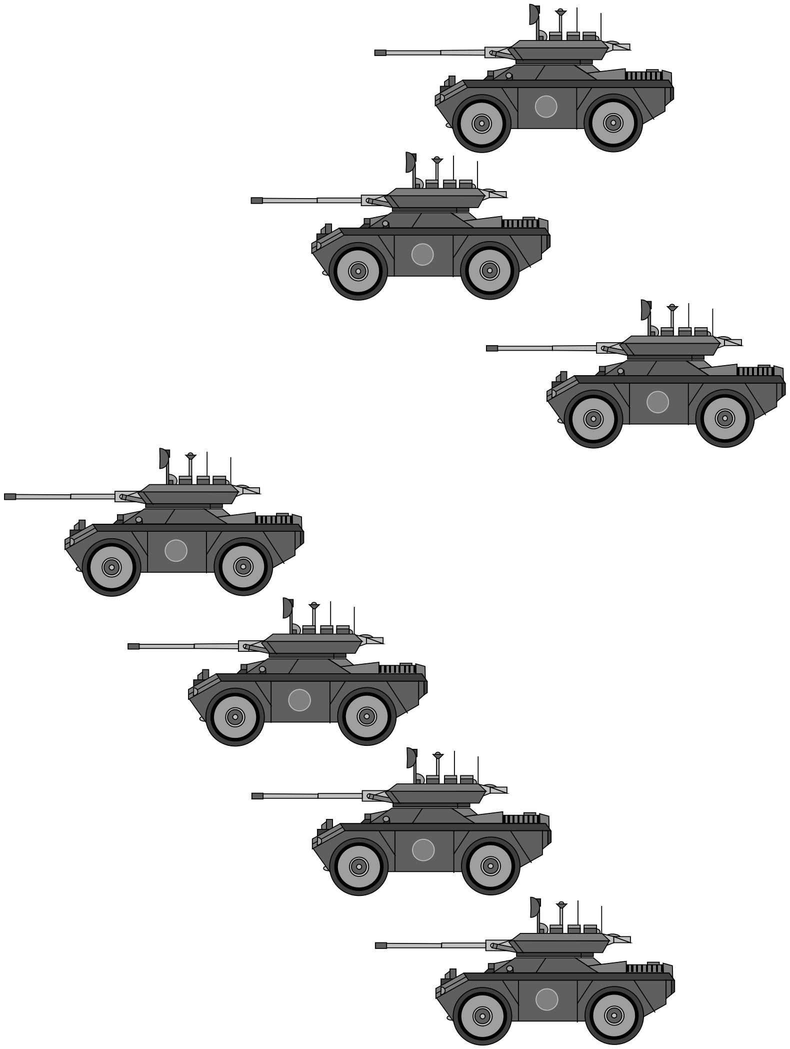
# What is Wrong With This Simulated Platoon?



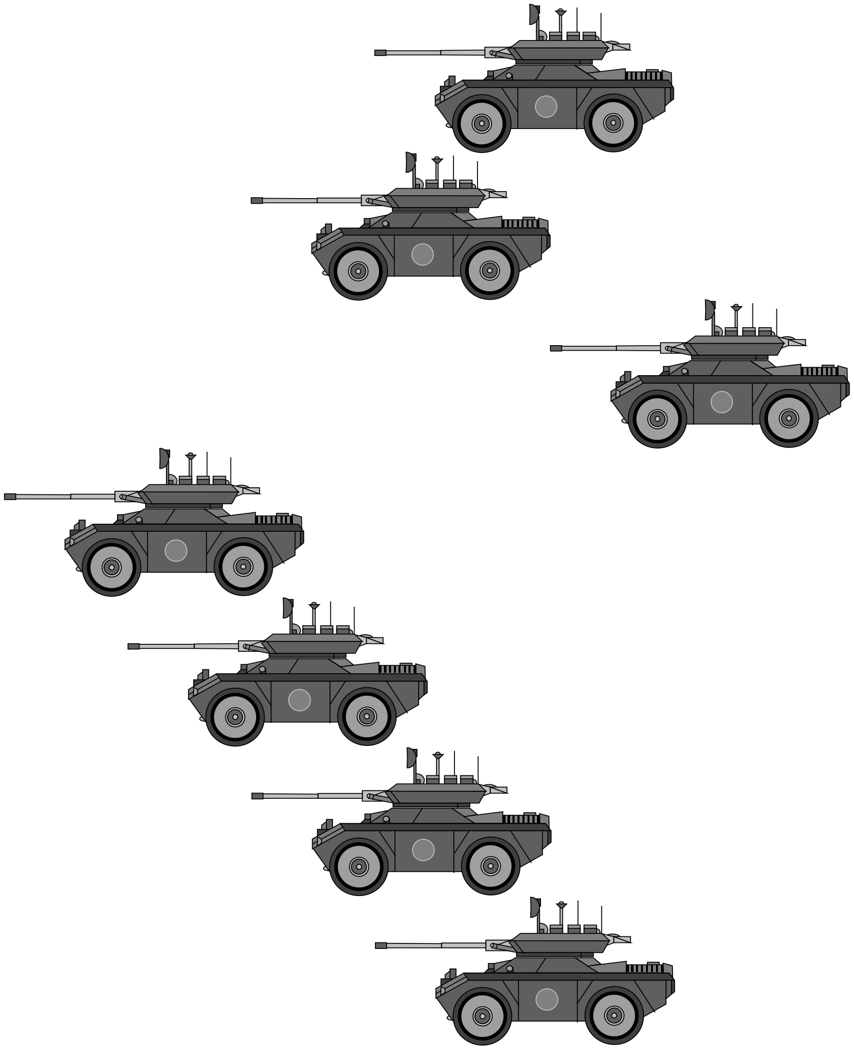
# What is Wrong With This Simulated Platoon?



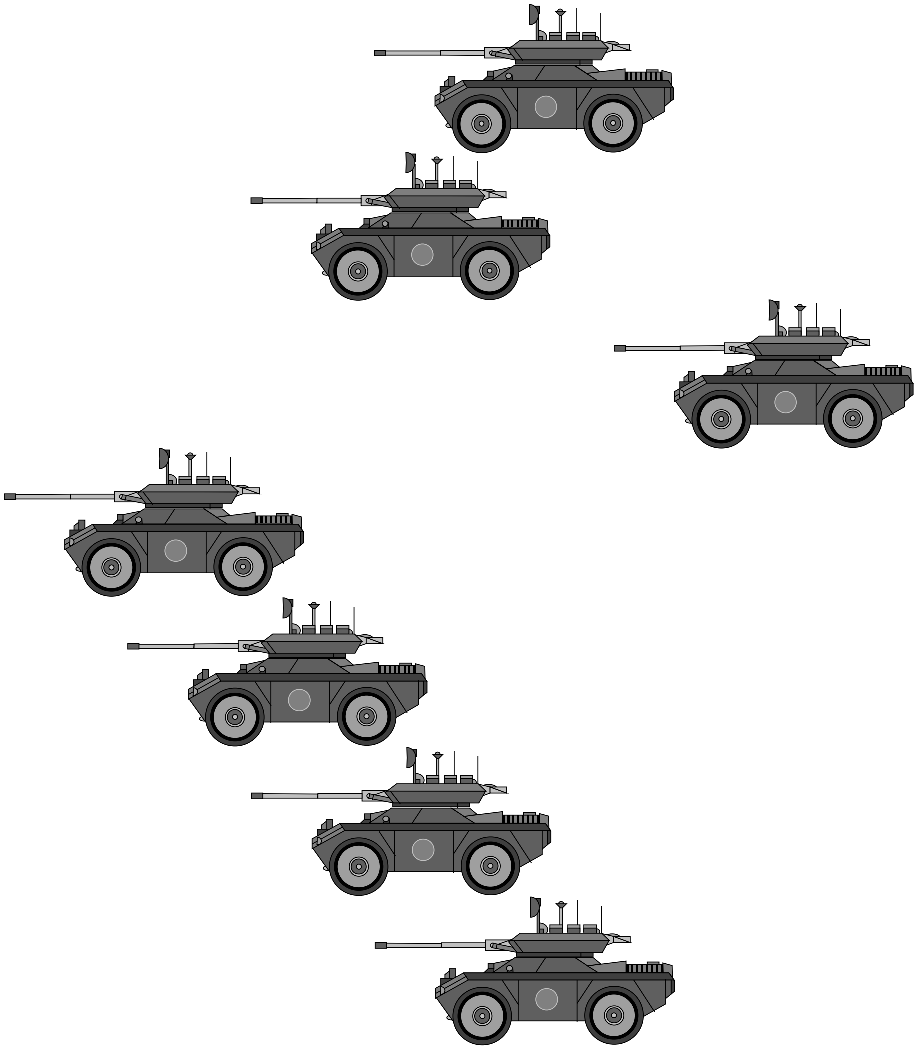
# What is Wrong With This Simulated Platoon?



# What is Wrong With This Simulated Platoon?



# What is Wrong With This Simulated Platoon?



# What is Wrong With This Simulated Platoon?

- The platoon disbanded just as this vehicle was joining up.
- Mismatched clocks meant that different parts of the simulation couldn’t agree on if/when the vehicle joined the platoon.

# What is Wrong With This Simulated Platoon?

- The platoon disbanded just as this vehicle was joining up.
- Mismatched clocks meant that different parts of the simulation couldn't agree on if/when the vehicle joined the platoon.

## Other errors were identified:

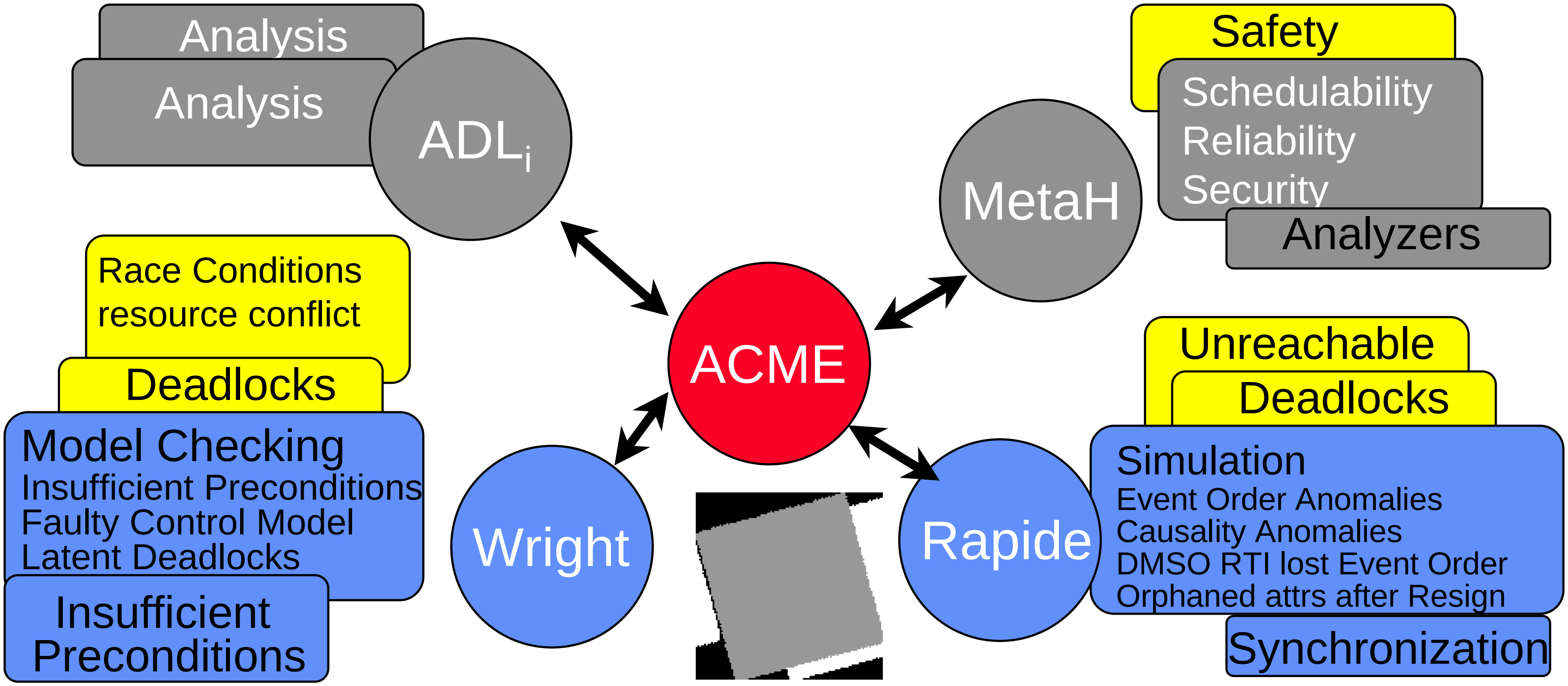
- The system loses information when the simulation is paused.
- Various cases occur where the system waits for events that will never occur (deadlock condition).

**These potential causes were all discovered by Architecture Analysis**

# Multiple Faults? Multiple Analyses!

## ACME developed as common interchange mechanism

- Supports common static analysis services
- Provides tool access through ADL translation
- Facilitates development of domain-specific notations



Analysis

Analysis

$ADL_i$

Safety

Schedulability
Reliability
Security

Analyzers

MetaH

Race Conditions
resource conflict

Deadlocks

ACME

Model Checking
Insufficient Preconditions
Faulty Control Model
Latent Deadlocks

Insufficient Preconditions

Wright

Rapide

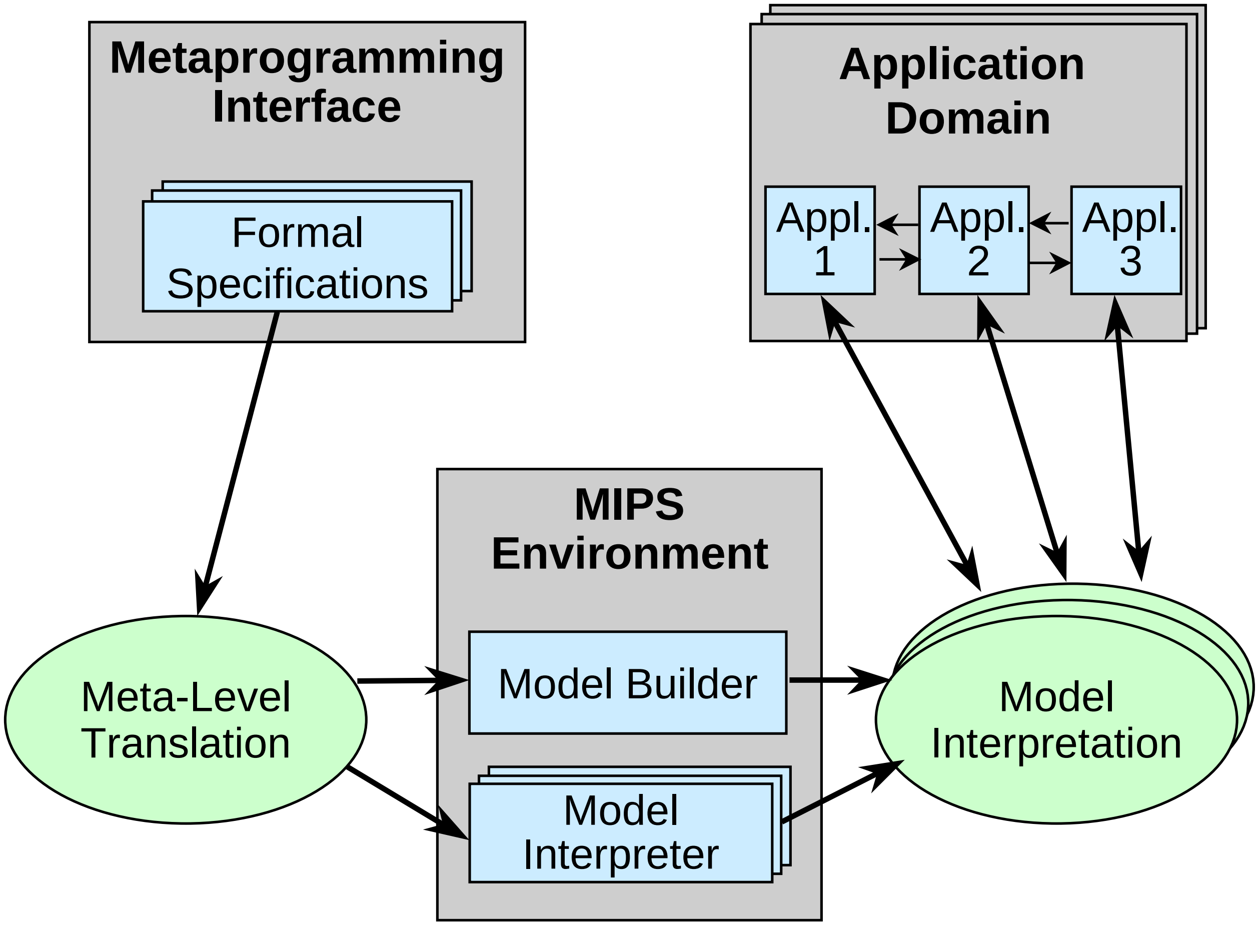
Unreachable

Deadlocks

Simulation
Event Order Anomalies
Causality Anomalies
DMSO RTI lost Event Order
Orphaned attrs after Resign

Synchronization

# Automated Composition and Generation



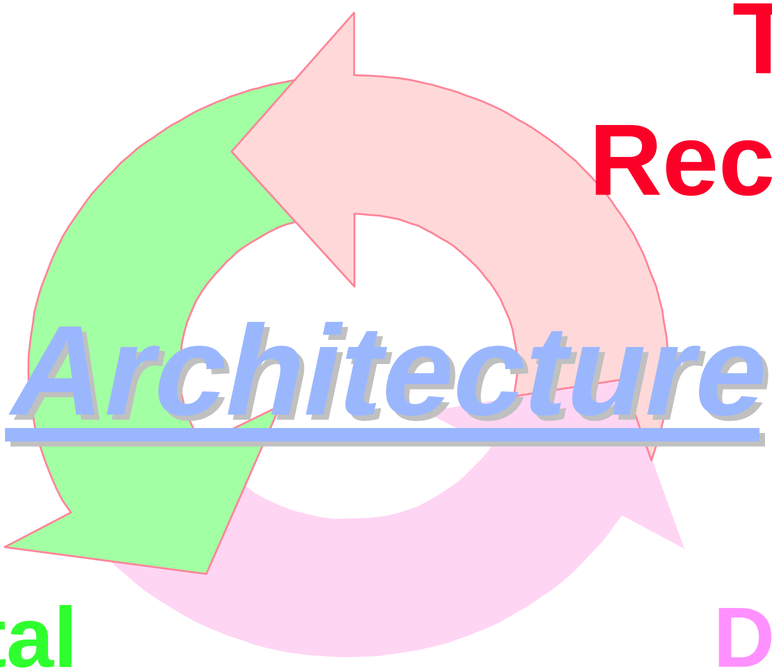
**Impact**

- Automated component composition, **not custom coding**
- Shift engineering focus to designing for change
- Months of setup time reduced to hours
- AEDC estimated 1996 return – **$10M**

# EDCS Technical Approach

Design Management

Test and Recertification



Architecture

Incremental change incurs incremental cost!

Disciplined Construction

# Test and Recertification

- **Requirement:**
  - Test effort proportional to size of change

One-net

- **Problem Today:**
  - Massive retest is expensive and time consuming
  - Testing is blind (usually started from scratch – can’t use analysis results or test history)

One-net

- **EDCS Approach:**
  - Specification-based Test Selection and Test Oracles
    - » **Static Dependency Analysis**
  - Regression testing based on test history and architectural analysis
  - Automated test instrumentation

# Test and Recertification

## Test and Analysis Tool Transition

- QuEST supported by Motorola, CDI, SW Bell, Motorola, Nortel, Lockheed, Hughes, SAIC
- Quest v2.0 Toolset Delivered, includes C/C++ processing components
- Conducting annual Commercial Tools and State-of-Practice Surveys
- Work underway to integrate NRL Formal Requirements tools

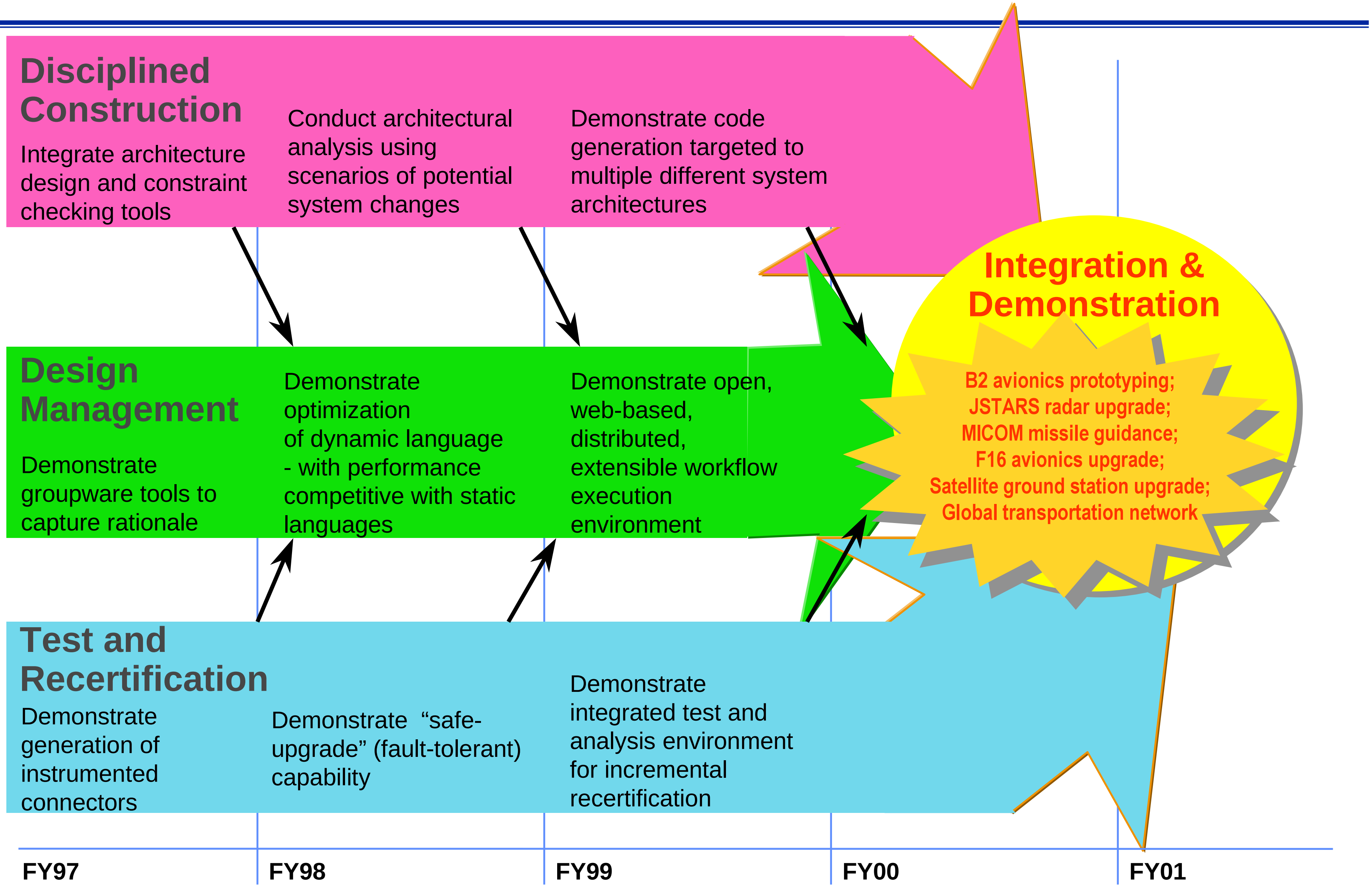
Package

## Safe Upgrade Capability

- Make changes to real-time software safer, in spite of potential (new) errors
- Scenario: Users need to upgrade baseline system to incorporate, e.g.,:
  - new functionality (e.g., Automated Maneuver and Attack System - AMAS algorithm, introduction of GPS)
  - algorithmic improvements
- Approach – analytically redundant modules generated from architectural specifications

Package

# Roadmap



## Disciplined Construction

Integrate architecture design and constraint checking tools

Conduct architectural analysis using scenarios of potential system changes

Demonstrate code generation targeted to multiple different system architectures

## Design Management

Demonstrate groupware tools to capture rationale

Demonstrate optimization of dynamic language - with performance competitive with static languages

Demonstrate open, web-based, distributed, extensible workflow execution environment

## Test and Recertification

Demonstrate generation of instrumented connectors

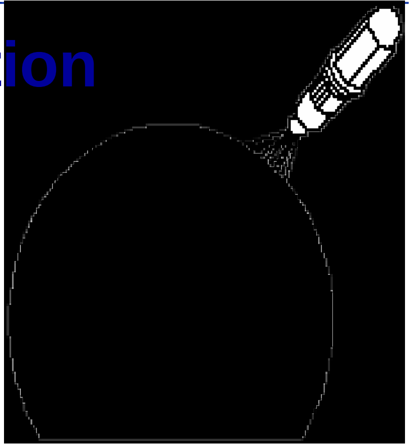
Demonstrate "safe-upgrade" (fault-tolerant) capability

Demonstrate integrated test and analysis environment for incremental recertification

## Integration & Demonstration

B2 avionics prototyping;
JSTARS radar upgrade;
MICOM missile guidance;
F16 avionics upgrade;
Satellite ground station upgrade;
Global transportation network

# EDCS Technology Demonstrations



## Satellite Control Network/Satellite Ground Station (SCN/SGS) upgrade: integrate an additional satellite and its data [USC/CSE, TRW & Aerospace]

- Capture design rationale
- Analyze alternative architectures
- Modify architectures using design rationale

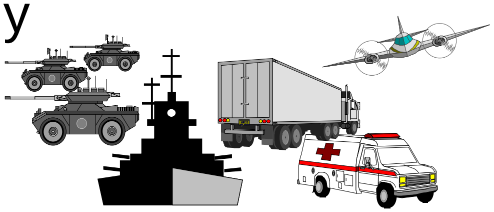
## Upgrade B2 software to support in-flight mission planning, build simulation environment [Northrop/Grumman]

- Use rapid prototyping
- Demonstrate architecture-level analysis, software understanding and visualization
- Conduct incremental test and certification

# EDCS Technology Demonstrations

## U.S. Transportation Command (USTRANSCOM) Global Transportation Network (GTN) [Lockheed Martin]

- WWW-based access to information synthesized from life-cycle artifacts
- Process, project, and product management capabilities
- Configuration management across geographically distributed sites
- Integration/evaluation of architectural description languages and tools



## E-8C Joint Surveillance Target Attack Radar System (JSTARS) [Modus Operandi, Northrop, USC, USC-ISI]

- CORBA-based infrastructure for system representation and analysis
- Requirements negotiation and management
- Intelligent documentation
- Legacy data integration

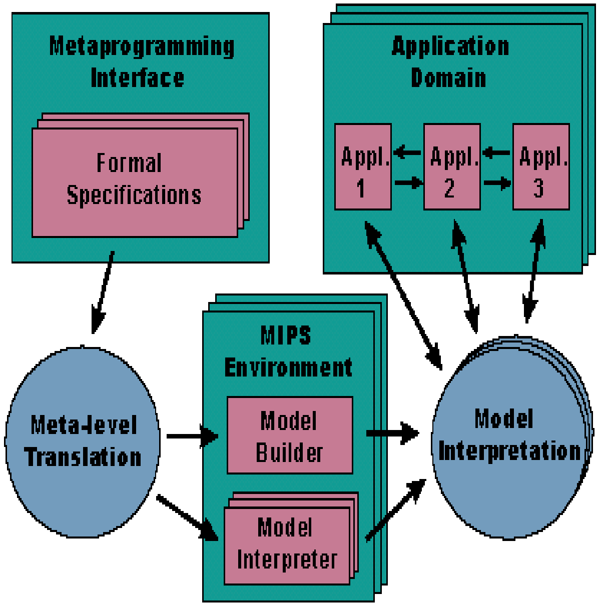
Approved for Public Release, Distribution Unlimited

# EDCS Technology Demonstrations

## High Confidence F-16 Avionics Upgrade:

[Carnegie Mellon University, SEI, & Lockheed Martin]

- INSERT technology for safe upgrades (fault tolerance)
- Formal architecture descriptions (using Honeywell's MetaH)
- Test and analysis
- Code synthesis



## Integrated Test Information System:

[Vanderbilt University, Arnold Engineering Development Center]

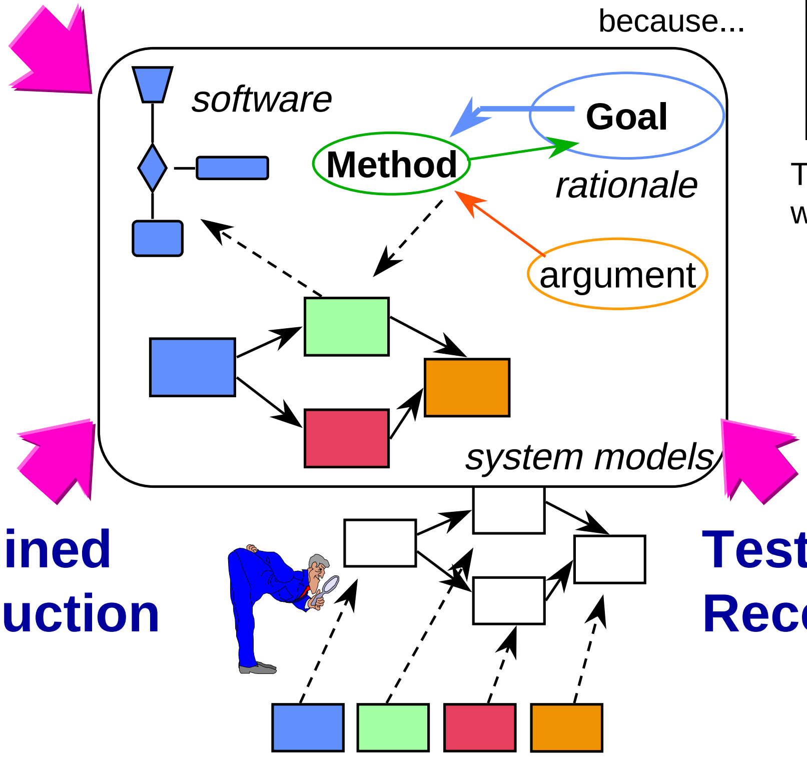
- Domain specific modeling and program synthesis environments
- CORBA-based integration of system design and access to test data (Catalyst)
- Dynamic Probes
- Java program generators

*Approved for Public Release, Distribution Unlimited*

# Vision

*Develop the technologies needed for continuous evolution of families of long-lived military software systems with costs proportional to the size of the change*

## Design Management



## Disciplined Construction

## Test & Recertification

Do it this way because...

That won't work because...

### Environment

- Long system lifetimes
- Changing missions
- Loss of design rationale
- Languages & tools sacrifice flexibility for efficiency
- Commercial sector focus on high-volume, modest reliability

# Architecture / Construction

## Architecture describes :

- component **topology** and **interactions**
- in terms of legal and illegal **configurations** and **sequences of events**.

EDCS is adding notions of **constraints, dynamic configurations**, and **standard representation.**

**It is particularly useful for errors related to event sequences – these cause deadlocks, lost data, erroneous results**